

UNIVERSITY OF ST ANDREWS

CS5014

PRACTICAL 1

Machine Learning

Author:
150008022

March 7, 2019



Goal

The goal of this practical was to cleanse and process real world data in order to produce a regression model, and evaluate its performance.

Contents

I	Loading and Cleaning the Data	1
II	Analysing and Visualising the Data	1
III	Feature Selection	5
IV	Selecting and Training a Regression Model	6
V	Performance Evaluation	7
VI	Discussion	7
VII	Conclusion	7

Part I

Loading and Cleaning the Data

Numpy was used to load the csv file. The header row was skipped, and to ensure there were no missing values, the invalid raise flag was also used when parsing the data. This would raise an exception if any rows were found to be missing data.

No preprocessing had occurred on the data set according to the original paper, and so the data could be used as given.

The input and output columns were separated into two variables `x` and `y`, in accordance with the notation used in lectures.

Next, the testing set was isolated from the available data. Simple random sampling (SRS) and stratified sampling were considered for performing the data splitting. SRS is intuitive, but for less uniformly distributed data sets, it can lead to subsets that poorly represent the input data, and therefore suffer from sampling bias [1].

For these reasons, stratified random sampling was used with a 80%-20% split. To ensure our training and testing sets were representative of our data, the output variables were used to categorise the data into strata. Since the two appeared to have a linear relationship as shown in figure 5, we could assume that using one column for categorising our data would produce an equal distribution of the values in both columns.

The 586 possible values of `Y1` were split into 50 bins using `numpy.digitize`, and then passed to `train_test_split` as the `stratify` argument.

Part II

Analysing and Visualising the Data

Firstly, a histogram of each of the input variables and outputs were plotted in order to visualise the distribution of the values (Figures 1 and 2). When comparing to the histograms from the given paper [2], most of the plots matched. Any differences were identified to be caused by 10 bins always

being used (the default if not specified by `numpy.hist`), and the paper would sometimes use more. However, it was still clear that none of the variables had a gaussian distribution. The output variables also appeared tail heavy, but the inputs did not.

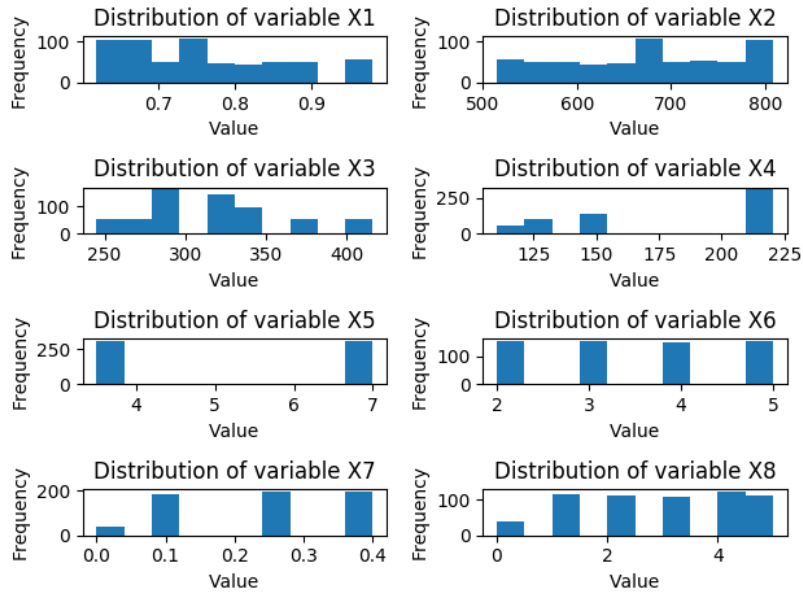


Figure 1: Distribution of input variables

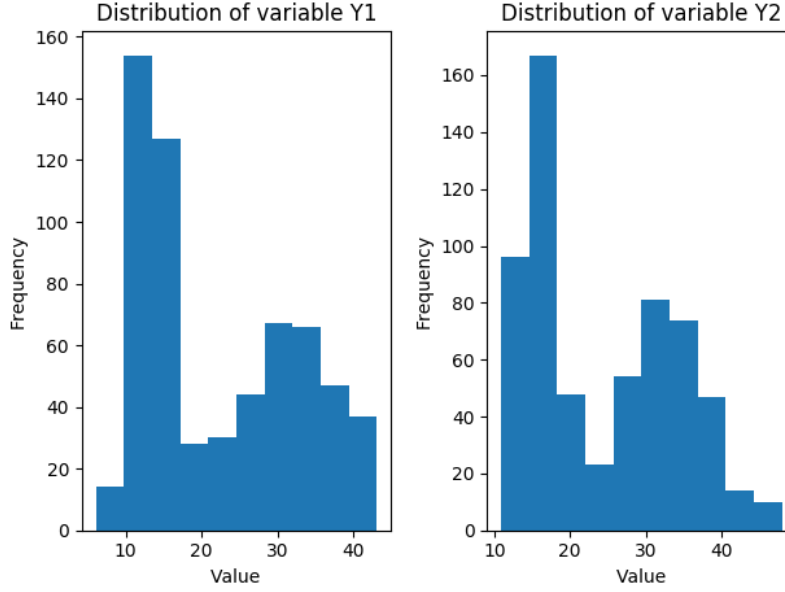


Figure 2: Distribution of output variables

In order to identify which variables had the strongest relationships with the outputs, and if these relationships were linear or monotonic, scatter graphs were made between each input and output variable (Figures 3 and 4). Feature scaling (specifically mean normalisation) was used to allow for comparisons between values that could have very different ranges. The resulting plots resembled those from the given paper, with different x-axis values since mean normalisation gave values in the range $-0.5 \leq x \leq 0.5$. To better visualise the density of the data points as well as their position, the `alpha` parameter was set to 0.1 in the plots.

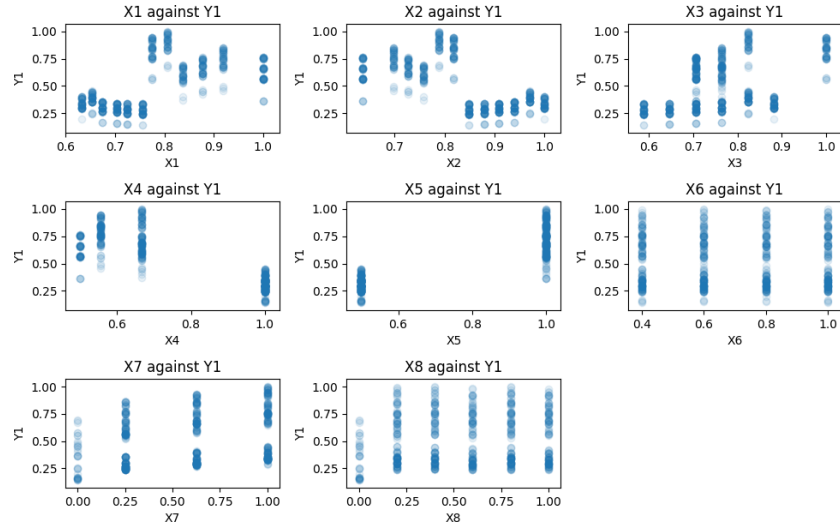


Figure 3: Normalised inputs plotted against the first output variable

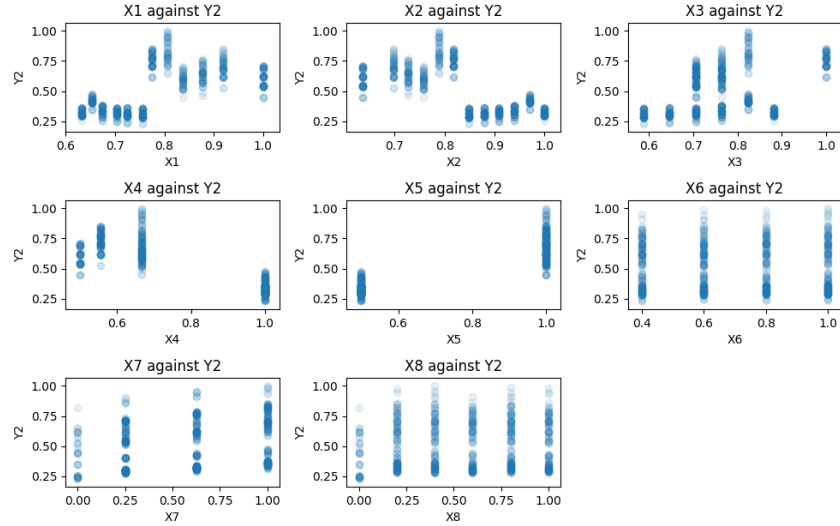


Figure 4: Normalised inputs plotted against the second output variable

Since the distribution of the outputs were so similar, they were also plotted against each other in a scatter plot (Figure 5). This showed that the two variables had very similar values, and so a regression model that applied to only one of them could likely be used for the other.

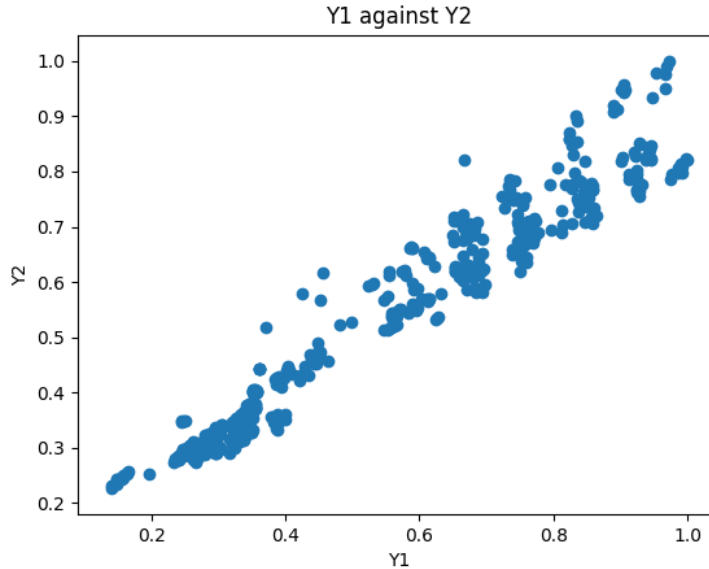


Figure 5: Normalised outputs against each other

Part III

Feature Selection

To try and identify which features had the strongest effect on the outputs, both the Pearson and Spearman rank correlation coefficients were considered. It was noted that the Pearson correlation would give a perfect value when the two variables were linearly related, whilst the Spearman correlation (a similiar alternative, and the one used in the original paper) would give a perfect value when the variables were monotonically related. Pearson is meant for use with continuous variables, whilst Spearman can be used for both continuous and ordinal variables, which our data set contains. Given these factors, the Spearman rank correlation coefficient was used as a filter method.

From the scatter plots, some variables appeared to have a possible linear relationship with the outputs (for example, X7 and Y1), whilst others had a monotonic relationship (for example, X1 and Y1). Using the Scipy *stats.spearmanr* method, the correlation coefficients were easily calculated,

alongside a p-value, as shown in table 1. Immediately from these result we can see that X6 and X8 show little evidence of a monotonic relationship existing between them and either of the outputs. This can also be seen from the scatter plots.

Given the number of input features, using them all would have negative effects on the choosen algorithm. Including each feature could result in overfitting, and would require more training data to eliminate this issue. It would also increase the computation time required to train our model. Features that have no real effect on the outputs would also act simply as noise, which would negatively effect our model. And since the amount of available data is small, it would be preferable to use fewer features, therefore X6 and X8 were removed from dataset.

Given more time and better knowledge of the subject matter, it may be possible to merge correlated inputs instead of remove them in order to reduce the number of dimensions of our input.

Table 1: Spearman rank correlation coefficients, with p values

X	Y	Rho	p
1	1	0.61	0.00
1	2	0.64	0.00
2	1	-0.61	0.00
2	2	-0.64	0.00
3	1	0.49	0.00
3	2	0.43	0.00
4	1	-0.80	0.00
4	2	-0.80	0.00
5	1	0.86	0.00
5	2	0.86	0.00
6	1	0.00	0.99
6	2	0.03	0.51
7	1	0.35	0.00
7	2	0.32	0.00
8	1	0.09	0.03
8	2	0.06	0.14

Part IV

Selecting and Training a Regression Model

The limited size of the corpus available made the choice of algorithm especially crucial. The lack of noticeable outliers in the visualisations at least suggest that the data is not of poor-quality. Due to the small corpus size, K-fold cross-validation was chosen to estimate the model error [3].

Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) were both considered as performance measures for regression. RMSE is more sensitive to outliers, but is generally preferred when absent, which is true for our input, and so it was chosen.

Part V

Performance Evaluation

Part VI

Discussion

Part VII

Conclusion

References

- [1] Z. Reitermanova. *Data Splitting*. Matfyzpress, 2010.
- [2] Athanasios Tsanas and Angeliki Xifara. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49:560–567, 2012.

- [3] Aurlien Gron. *Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow*. O'Reilly Media, 2018.